A High-Level Approach Towards End User Development in the IoT

A High-Level Approach Towards End User Development in the IoT

Fulvio Corno
Politecnico di Torino
Corso Duca degli Abruzzi, 24
Torino, Italy 10129
fulvio.corno@polito.it

Luigi De Russis
Politecnico di Torino
Corso Duca degli Abruzzi, 24
Torino, Italy 10129
luigi.derussis@polito.it

Alberto Monge Roffarello
Politecnico di Torino
Corso Duca degli Abruzzi, 24
Torino, Italy 10129
alberto.monge@polito.it

CHI'17 Extended Abstracts, May 6–11, 2017, Denver, CO, USA.
ACM ISBN 978-1-4503-4656-6/17/05.
http://dx.doi.org/10.1145/3027063.3053157

Abstract
Programming environments for end-user personalization in the Internet of Things (IoT) are becoming increasingly common. They allow users to define simple IoT applications, i.e., connections between different IoT devices and services. Unfortunately, the adopted representation models are highly technology-dependent, e.g., they often categorize devices and services by manufacturer or brand. Such an approach is not suitable to face the expected growth of the IoT, nor it allows to adapt to yet undiscovered IoT services. In this paper, we present a generic and technology-independent representation for IoT end-user programming environments. The aim of this "high-level" representation is to allow end-users to create abstract IoT applications that adapt to different contextual situations. We preliminary evaluated the representation by comparing it with the one used by existing programming environments in a user study with 10 participants. Results show that the representation is understandable, and it allows users to create IoT applications more correctly and quickly.

Author Keywords
End User Development; Internet of Things; Trigger-Action

ACM Classification Keywords
H.5.m [Information interfaces and presentation (e.g.,HCI)]: Miscellaneous

Figure 1: A small portion of devices and services modeled by IFTTT. Each technology has its own different element, with different triggers and actions inside. With the spread of new smart "things", the amount of information may become too high and the interface cluttered.

Introduction and Motivations

The Internet of Things (IoT) is a recognized paradigm that may help society in many different ways, i.e., with applications for the individual, the planet, and industries as well [2]. However, given the rapid growth of the number of "smart" objects [6], the increasing complexity of the IoT ecosystem raises new challenges, especially in the interaction with end users. Since IoT solutions often adopt different standards and technologies, the question of interoperability between smart devices and services still remains open, as already reported by Munjin [9]. In this context, End User Development (EUD) environments enable end-users with and without programming skills to customize their own IoT objects on the basis of their personal needs. By following this trend, third-party services for end-user personalization such as IFTTT[1], Atooma[2], and Tasker[3] are becoming increasingly common. Such programming environments typically employ the trigger-action programming paradigm (e.g., *"if something happens, then perform an action"*) to allow the definition of IoT applications (often named *rules*), i.e., connections between pairs of IoT devices and/or services. Unfortunately, they present some limits, that *John* has widely experienced:

John, a manager of an important company, is always hot, especially in summer. He loves air conditioning, and he would like to set a low temperature wherever it is possible. At home, John has an intelligent Nest thermostat that he controls through his Android smartphone. John goes to work by car. There, all the offices are equipped with a Samsung smart air conditioner.

John has to define several rules to reach his comfort goal, at least one for his home, one for his office, and one for his car, even if they perform the same logical operations (i.e., set a specific temperature when he enters a place). Furthermore, he has to be aware of every single technology he may encounter before creating his rules (e.g., Nest, Samsung, etc.), to choose the right one for each rule. Finally, even with an authorization, *John* will not be able to define similar rules for unknown places or "things" (e.g., his friend's car). Such three major issues (high number of rules required, no technology awareness, and no discovery) are due to the intrinsic technology-dependency of the representation models adopted by many existing end-user programming environments in the IoT. In the case of IFTTT, for example, devices and services are simply grouped by manufacturer or brand (Figure 1). To overcome the issues, a new breed of programming environments should be designed to support a "higher level" representation of IoT devices and services.

To take a step towards this direction, the contribution of this work is a **high-level representation**, named EupONT for IoT EUD environments. Such a representation allows *John* to define a single rule for his need, e.g., *"if I enter a closed space, then set the temperature to 20 Celsius degree"*. The representation abstracts the IoT ecosystem by modeling devices and services on the basis of their functionality and capabilities. It allows the definition of generic and technology-independent trigger-action rules which can be adapted to different contextual situations, independently of manufacturers, brands, and other technology-related details. EupONT has been defined as an ontological framework, thus providing semantic and reasoning capabilities for supporting the run-time execution of the definable end-user rules. For a preliminary evaluation, we focused on rules composition and we compared the EupONT representa-

[1] http://ifttt.com (last visited on November 11, 2016)
[2] http://www.atooma.com (last visited on November 11, 2016)
[3] http://tasker.dinglisch.net (last visited on November 11, 2016)

tion with the one adopted by IFTTT. In the in-lab study with 10 participants, we investigated whether and how EupONT helps users in the composition of trigger-action rules. Results show that, with the new representation, users can create IoT applications with fewer errors and in less time.

Background and Related Work

Lieberman et al. [8] defines End User Developement as *"a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artifact"*. Starting from iCAP [5], there has been a long history of interest in EUD and mashup technologies. EUD is, nowadays, a promising approach even for the IoT. The spread of interconnected devices and services allow users to effectively participate in the IoT [3], and people are increasingly moving from passive consumers to active producers of both information, data, and software [9]. Several commercial tools (e.g., IFTTT, Atooma, and Tasker) allow users to customize the joint behavior of their IoT devices and services. The concepts used by such tools have also been extended for smart homes [10, 4], and for cross-device interfaces [7].

Past and recent solutions in End User Development mainly use the same programming paradigm, i.e., trigger-action programming, that offers a very simple solution to create IoT applications. Furthermore, trigger-action rules provide a good match with users mental model [5]. However, recent studies [1, 10] show that the trigger-action approach should be further investigated to cope with the evolving IoT world. As reported by Barricelli and Valtolina [1], the widespread growth of IoT devices and services influences the most well established definitions of EUD. They extended the trigger-action paradigm in the IoT by incorporating new concepts to those already modeled (e.g., other IoT users, space and time, and recommendation systems). Furthermore, Ur et al. [10] discovered that users would like to express triggers with a higher level of abstraction than the one offered by contemporary IoT end-user programming environments. Starting from these evidences, our work tries to abstract the end-user development in the IoT with a new representation of devices and services, the EupONT high-level representation.

High-Level Representation for EUD

EupONT is a model representation for EUD that allows the composition of generic and technology-independent IoT applications, which can be adapted to different contextual situations, independently of manufacturers, brands, and other technology related details. We designed EupONT as a semantic model, thus exploiting the capabilities offered by ontologies. With a semantic model, we can easily perform queries on the representation such as *"which IoT devices or services can perform a particular action?"* or *"which IoT devices or services can generate a particular event?"*. We firstly defined the main concepts to be inserted in the ontology[4] (e.g., *Trigger*, *Action*) and their relationships. Then, in order to define a taxonomy for each main general concepts, we asked two expert HCI researchers in the field of EUD and IoT to analyze the entire ecosystem modeled by IFTTT. The primary reasons for selecting IFTTT include its popularity, the fact that it is freely available, and the availability of a large repository of rules [11]. Thanks to the analysis, an initial set of relevant terms was extracted. Such terms were then hierarchical organized, for example, to group together devices and services with the same final capabilities. The coding process was performed independently by the two HCI researchers. They compared their results until they reached a 80% agreement.

[4] The ontology is available at http://elite.polito.it/ontologies/eupont.owl

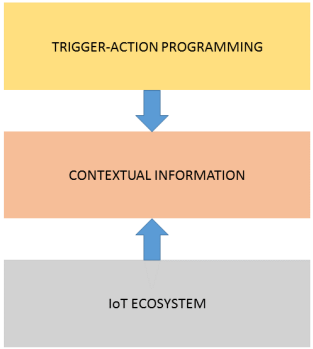

Figure 2: The structure of the high-level representation. The two main layers of the representation (i.e., trigger-action programming and IoT ecosystem) refer to the same contextual information.

Description
The general structure of EupONT is shown in Figure 2. The two main layers of the representation are the **trigger-action programming** and the **IoT ecosystem**. They refer to the same **contextual information** layer, which describes locations and users of the modeled IoT ecosystem.

The **trigger-action programming** layer describes our trigger-action programming approach. It extends the approaches adopted by existing IoT programming environments since it allows the definition of more generic and technology-independent rules. Triggers and actions are classified in 9 categories: *Transportation*, *Place*, *Temperature*, *Physical Exercise*, *Lighting*, *Communication*, *People Availability*, *Humidity*, and *News*. In this way, if a user is interested in controlling the temperature of an environment, she can use the triggers and the actions of the *Temperature* category. Similarly, the *Communication* category offers abstract actions and triggers (e.g., *"send a message"*) to allow the communication with other users. Triggers and actions are hierarchical organized in such a way the user could decide her preferred level of abstraction. For example, Figure 3 shows some definable *actions* related of the *Lighting* category. The action to increase the lighting in a place (*Illuminate*) may include, for instance, turning the lights on, or opening the blinds. For each trigger and action, we include the possibility of defining three optional restrictions:

- *Who:* the user(s) involved by the trigger or the action. User(s) can be generic (e.g., *"any person"*) or specific (e.g., *"my friend Mark"*).
- *Where:* the place where the trigger has to be registered, or where the action has to be performed. Locations can be generic (e.g., *"any building"*) or specific (e.g., *"my home"*).

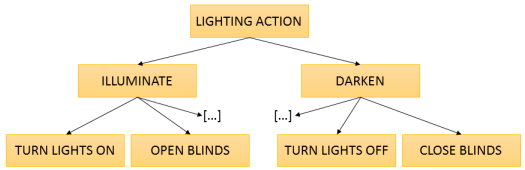

Figure 3: A partial view of the hierarchical tree that characterizes lighting-related actions.

- *Value:* a series of one or more values related to the action or the trigger (e.g., a temperature threshold to monitor, or the brightness level to set).

The **IoT ecosystem model** layer models IoT devices and services on the basis of their categories (e.g., lighting systems, user devices, smart appliances) and their final capabilities (e.g., switching, sensing, actuating, communication). In particular, devices and web services are often composite or complex, thus offering several capabilities. With a smartphone, for example, you can trace your position through the GPS, send messages, connect to a Bluetooth device, etc. To take into account this complexity, we modeled devices and services as a set of objects able to expose one or more functionality. Each functionality describes a capability of an object, and may have commands to perform some actions, or notifications to register event listeners. Commands and notifications include the features needed to interact with the specific technology (e.g., specific commands with required parameters to be sent to devices and services). Through an automatic mechanism based on a set of predefined SWRL rules[5], each IoT device or service is linked with actions and triggers it can serve. For example, a lighting system (e.g.,

[5] https://www.w3.org/Submission/SWRL/ (last visited on December 27, 2016)

a lamp) that offers a switching-on capability is able to reproduce a *"turn lights on"* action. Furthermore, thanks to the hierarchical organization of triggers and actions, a system able to *turn lights on* is also able to reproduce the more generic behavior of the hierarchy, i.e., *illuminate* a place.

The **contextual information** describes locations and users that act as attributes for triggers and actions (*Where* and *Who*), and define the context information of the IoT ecosystem. Thanks to this layer, IoT applications can be adapted in run-time to different users and (even unknown) locations. In this way, the ontological representation provides a strong support for executing the defined end-user rules.

Preliminary Evaluation

Since IoT applications are intended to be composed by end-users, the model of EupONT should be highly comprehensible, at least for what concerns the trigger-action programming layer. For this reason, in the evaluation we focused on the rule composition, leaving for future works their actual execution. In particular, we compared our high-level representation with the representation model used by IFTTT. We were interested in evaluating a) whether the high-level representation is understandable at least as the model employed by IFTTT, and b) whether the high-level representation allows to create IoT applications more efficiently (i.e., in less time) than IFTTT.

Study Design

We carried out a controlled in-lab study with 10 participants (4 female) with a mean age of 22.40 years (SD = 2.46). To consider both users with and without formal programming training, we recruited 5 participants from the Department of Control and Computer Engineering of our university, and 5 participants from the school of psychology of the University of Turin. To allow the comparison between the two

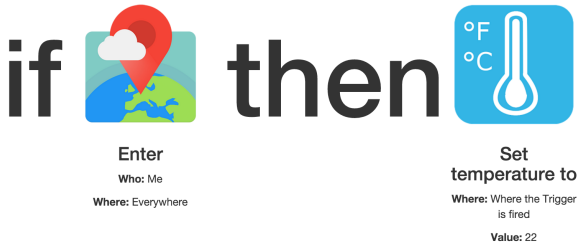

Figure 4: Our prototype interface in the high-level representation, showing the rule "If I enter any place, then set the temperature to 22 Celsius degree"

approaches, we created an interface modeled after IFTTT in two versions. One version allowed the composition of trigger-action rules in EupONT (Figure 4), while the other version exploited the low-level model, cloned by IFTTT. The experiment was a trial of 5 scenario-based tasks related to the creation of IoT applications. Each task consisted of two different parts: a *user scenario*, and a *goal*. The user scenario described a generic user, her owned devices, and some of her typical activities. The goal defined a specific behavior that the user wanted to obtain from her devices, and it was definable with one or more trigger-action rules. An example of a task was:

User scenario: *Mary is a researcher that works for an important university. She is interested in saving energy. However, she is often distracted, and she always forgets to turn the lights off. For this reason, she equipped her bedroom with home with two two Philips Hue lamps, and her living room with a Stack Lighting lamp. Furthermore, she installed a Samsung SmartThings Hub to remotely control the doors and the surveillance system of her house. Also her office is equipped with smart devices, e.g., LFIX smart lights.*

Goal: *Mary would like a way to automatically turn the lights off when she leaves a rooms or her office.*

Participants completed each task twice, with EupONT and with the IFTTT-like representation. The order of the tasks and the used representations were counterbalanced. We manually assigned a correctness level (from 0 to 100) to each task completed in the two representations. We assigned 0 points for totally wrong tasks, 50 points for partially correct tasks (e.g., tasks completed with some missed rules), and 100 points for correct tasks (i.e., tasks completed with a set of rules compliant with the scenario that exactly reproduced the goal). Furthermore, we measured the time needed by the participants to carry out each task.

Results

To evaluate whether EupONT is at least understandable as the model employed by IFTTT, we performed a one-way ANOVA in SPSS by considering the correctness level as dependent variable, and the used representation as the within-subject independent variable. Results show that the used representation significantly influenced the correctness level of the tasks ($F(1,9) = 7.83, p < .05$). In particular, the correctness level was higher with the high-level representation that with the representation adopted by IFTTT ($M = 82.00, SD = 5.33$ vs $M = 62.00$, $SD = 7.42$, respectively). Post-hoc test with Bonferroni correction revealed that this difference was statistically significant ($p < .05$), thus demonstrating that EupONT allows users to create IoT applications more correctly.

To evaluate whether EupONT allows users to create IoT applications more efficiently than IFTTT, we performed the same analysis by considering the average task duration in each representation as dependent variable. Results show that the used representation significantly influenced the time needed by the participants to carried out the tasks ($F(1,9) = 6.01, p < .05$). In particular, the task duration was lower with the high-level representation that with the low-level one adopted by IFTTT ($M = 96s, SD = 11s$ vs $M = 143s, SD = 14s$, respectively). Post-hoc test with Bonferroni correction revealed that this difference was statistically significant ($p < .05$), thus demonstrating that EupONT allows users to create IoT applications more efficiently.

Our first results suggest that the high-level representation is well understood by end-users and it is suitable for creating IoT applications. In fact, the new representation improved the correctness of the tasks carried out by the participants, thus facilitating users to define their rules. In addition, as expected, the high-level representation allowed the participants to complete the tasks in less time.

Conclusion and Future Works

In this work, we introduced a new approach towards end-end user development in the IoT, i.e., a high-level representation to be used by IoT programming environments. Such a representation, named EupONT, allows end-users to define generic and technology-independent trigger-action rules, that can be adapted to different contextual situations. EupONT has been preliminarily evaluated for rule composition with end-users. Results show that the high-level representation is understandable by end-users, and it may allow users to create IoT applications more effectively and efficiently than IFTTT, one of the most popular programming environments in this domain. Future work will better investigate the understandability and the usability of the high-level representation. Furthermore, by exploiting the ontological and reasoning capabilities of the high-level representation, we are in the process of developing a system that allows the execution of such trigger-action rules.

References

[1] Barbara Rita Barricelli and Stefano Valtolina. 2015. *End-User Development: 5th International Symposium, IS-EUD 2015, Madrid, Spain, May 26-29, 2015. Proceedings*. Springer International Publishing, Cham, Germany, Chapter Designing for End-User Development in the Internet of Things, 9–24. DOI: http://dx.doi.org/10.1007/978-3-319-18425-8_2

[2] Vint Cerf and Max Senges. 2016. Taking the Internet to the Next Physical Level. *IEEE Computer* 49, 2 (Feb 2016), 80–86. DOI: http://dx.doi.org/10.1109/MC.2016.51

[3] Jose Danado and Fabio Paternò. 2014. Puzzle: A mobile application development environment using a jigsaw metaphor. *Journal of Visual Languages & Computing* 25, 4 (2014), 297–315. DOI: http://dx.doi.org/10.1016/j.jvlc.2014.03.005

[4] Luigi De Russis and Fulvio Corno. 2015. HomeRules: A Tangible End-User Programming Interface for Smart Homes. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '15)*. ACM, New York, NY, USA, 2109–2114. DOI: http://dx.doi.org/10.1145/2702613.2732795

[5] Anind K. Dey, Timothy Sohn, Sara Streng, and Justin Kodama. 2006. iCAP: Interactive Prototyping of Context-aware Applications. In *Proceedings of the 4th International Conference on Pervasive Computing (PERVASIVE'06)*. Springer-Verlag, Berlin, Heidelberg, 254–271. DOI: http://dx.doi.org/10.1007/11748625_16

[6] Dave Evans. 2011. *The Internet of Things: How the Next Evolution of the Internet Is Changing Everything*. Technical Report. Cisco Internet Business Solutions Group.

[7] Giuseppe Ghiani, Marco Manca, and Fabio Paternò. 2015. Authoring Context-dependent Cross-device User Interfaces Based on Trigger/Action Rules. In *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia (MUM '15)*. ACM, New York, NY, USA, 313–322. DOI: http://dx.doi.org/10.1145/2836041.2836073

[8] Henry Lieberman, Fabio Paternò, Markus Klann, and Volker Wulf. 2006. *End User Development*. Springer Netherlands, Dordrecht, Netherlands, Chapter End-User Development: An Emerging Paradigm, 1–8. DOI: http://dx.doi.org/10.1007/1-4020-5386-X_1

[9] Dejan Munjin. 2013. *User Empowerment in the Internet of Things*. Ph.D. Dissertation. Université de Genève. http://archive-ouverte.unige.ch/unige:28951

[10] Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L. Littman. 2014. Practical Trigger-action Programming in the Smart Home. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 803–812. DOI: http://dx.doi.org/10.1145/2556288.2557420

[11] Blase Ur, Melwyn Pak Yong Ho, Stephen Brawner, Jiyun Lee, Sarah Mennicken, Noah Picard, Diane Schulze, and Michael L. Littman. 2016. Trigger-Action Programming in the Wild: An Analysis of 200,000 IFTTT Recipes. In *Proceedings of the 34rd Annual ACM Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3227–3231. DOI: http://dx.doi.org/10.1145/2858036.2858556